Near Optimal Multiple Sequence Alignments using a Traveling Salesman Problem approach

Chantal Korostensky and Gaston Gonnet
Swiss Federal Institute of Technology
Institute of Scientific Computing
ETH Zrich, Switzerland
korosten, gonnet@inf.ethz.ch

Abstract

We present a new method for the calculation of multiple sequence alignments (MSAs). The input to our problem are n protein sequences. We assume that the sequences are related with each other and that there exists some unknown evolutionary tree that corresponds to the MSA. One advantage of our method is that the scoring can be done with reference to this phylogenetic tree, even though the tree structure itself may remain unknown. Instead of computing an evolutionary tree, we only need to compute a circular tour of the tree which is determined via a Traveling Salesman Problem (TSP) algorithm. Our algorithm can calculate a near optimal MSA and has a performance guarantee of $\frac{n-1}{n} \cdot opt$ (where opt is the optimal score of the MSA). The algorithm runs in $O(k^2n^2)$ time, where k is the length of the longest input sequence. From there we improve the alignment further. Experimental results are shown at the end.

1. Introduction

The computation of multiple sequence alignments (MSAs) remains one of the major open problems in computational biology. Possibly this is due to the complexity as most problems associated with MSAs are NP-complete [20]. It is therefore desirable to develop heuristics that try to come as close to the optimum as possible but terminate within reasonable time and space bounds. An MSA consists of a set of strings, where each string represents an amino acid, DNA or RNA sequence (see Definition 1.1).

Example: In the MSA in Figure 1 five sequences are aligned ($n = 5$). We assume that the sequences are related and that we have the correct tree. The internal nodes represent unknown ancestor sequences.

1.0.1. Definitions

Definition 1.1 *Given is a set of sequences $S = \{s_1, .., s_n\}$ with $s_i \in \Sigma^*$ where Σ is a finite alphabet. A Multiple Sequence Alignment (MSA) consists of a set of sequences $\mathrm{A} = \langle a_1, a_2, .., a_n \rangle$ with $a_i \in \Sigma'^*$ where $\Sigma' = \Sigma \cup \{"_"\} \notin \Sigma$. $\forall a_i \in \mathrm{A} : |a_i| = k$. The sequence obtained from $a_i \in \mathrm{A}$ by removing all "_" gap characters is equal to s_i.*

Definition 1.2 *The character "_" or any contiguous sequence of "_" within an aligned sequence $a_i \in \mathrm{A}$ is called a gap. A gap g has length $len(g)$, where the length is the number of contiguous dashes "_". It starts at a position $pos(g)$ in sequence $a_i \in \mathrm{A}$ and ends at position $end(g)$.*

Definition 1.3 *An MSA scoring function is a function $F : \mathrm{A} \rightarrow I\!R$.*

Definition 1.4 *Let $\mathcal{A}$ be the set of all possible MSAs that can be generated for a given set of sequences $S = \{s_1, s_2, .., s_n\}$. The optimal MSA $\bar{\mathrm{A}} \in \mathcal{A}$ is an MSA such that w.l.o.g. $F(\bar{\mathrm{A}}) = \max_{\mathrm{A} \in \mathcal{A}} F(\mathrm{A})$. In many scoring functions the minimum is the optimal.*

Problem 1.1 (MSA problem) *Given is a set of sequences $S = \{s_1, .., s_n\}$. Find the optimal MSA A for S.*

1.0.2. Scoring Pairwise Sequence Alignments

To determine if two sequences $s_1, s_2 \in \Sigma$ are related and have a common ancestor the sequences are usually aligned, and the problem is to find the alignment that maximizes the probability that the two sequences are related:

To actually calculate these probabilities, one applies a Markovian model for sequence evolution [25, 3]. This begins with an alignment of the two sequences, e.g.

```
A: RPCVCP___VLRQAAQ__QVLQRQIIQGPQQLRRLF_AA
B: RPCACP___VLRQVVQ__QALQRQIIQGPQQLRRLF_AA
C: KPCLCPKQAAVKQAAH__QQLYQGQLQGPKQVRRAFRLL
D: KPCVCPRQLVLRQAAHLAQQLYQGQ_____RQVRRAF_VA
E: KPCVCPRQLVLRQAAH__QQLYQGQ_____RQVRRLF_AA
```

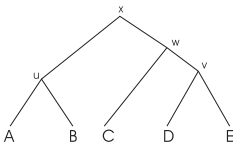


Figure 1. The sequences $\{A, B, C, D, E\}$ of the MSA on the left are related by the evolutionary tree on the right.

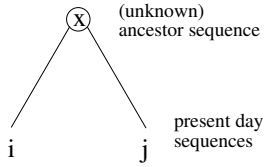
```
VNRLQQNIVSL_______________EVDHKVANYKP
VNRLQQSIVSLRDAFNDGTKLLEELDHRVLNYKP
```

The gaps arise from *insertions* (or their counterpart *deletions*) during divergent evolution. The alignment is normally done by a dynamic programming (DP) algorithm using Dayhoff matrices [12, 32, 28, 1], which finds the alignment that maximizes the probability that the two sequences evolved from an ancestral sequence as opposed to being random sequences. An affine gap cost is used according to the formula $a+l\cdot b$, where a is a fixed gap cost, l is the length of the gap and b is the incremental cost [1, 4]. More precisely, we are comparing two possibilities

a) that the two sequences arose independently of each other (implying that the alignment is entirely arbitrary, with amino acid i in one protein being aligned to amino acid j in the other is occurring no more frequently than expected by chance, which is equal to the product of the individual frequencies with which amino acids i and j occur in the database)

$$Pr\{i \text{ and } j \text{ are independent}\} = f_i f_j \qquad (1)$$

b) that the two sequences have evolved from some common ancestral sequence after t units of evolution where t is measured in PAM units [9].



$Pr\{i \text{ and } j \text{ descended from some } x\}$

$$= \sum_x f_x Pr\{x \to i\} Pr\{x \to j\} \qquad (2)$$

Definition 1.5 *The score of an optimal pairwise alignment $OPA(s_1, s_2)$ of two sequences s_1, s_2 is the score of an alignment with the maximum score where a probabilistic scoring method [6, 10] is used. We refer to a pairwise alignment of two sequences s_1, s_2 with $\langle s_1, s_2 \rangle$.*

$$D_{ij} = 10 \log_{10} \left(\frac{Pr\{i \text{ and } j \text{ related}\}}{Pr\{i \text{ and } j \text{ random}\}} \right) \qquad (3)$$

The entries of the Dayhoff matrix are the logarithm of the quotient of these two probabilities. Note that scores represent the probabilities that the two sequences have a common ancestor. The larger the score is the more likely it is that the two sequences are homologous and therefore have a common ancestor.

1.0.3. Related work

There are two main methods for finding an optimal MSA:

- global methods which construct an alignment throughout the length of the entire sequence (see below), and
- local methods which only attempt to identify an ordered series of motifs (homologous regions) while ignoring regions between motifs [17, 18].

We are concerned with global methods and we briefly describe the different strategies used along with some of their drawbacks.

1.0.4. Heuristics

Progressive Alignment: This approach begins with the alignment of the two most closely related sequences (as determined by pairwise analysis) and subsequently adds the next closest sequence or sequence group to this initial pair [37, 7]. This process continues in an iterative fashion, adjusting the positioning of indels in all sequences.

The major shortcoming of this approach is that a bias may be introduced in the inference of the ordered series of motifs (homologous parts) because of an overrepresentation of a subset of sequences.

Clustering Methods: Sequences are clustered into subalignments by using a similarity measure [33, 34, 26]. The clustered subalignments are subsequently aligned to one another by employing various consensus methods that reduce each subalignment to a single consensus sequence.

1.0.5. Exact Methods and Approximation Algorithms

Tree Alignment (TA) and Generalized Tree Alignment (GTA): An MSA is constructed with respect to a phylogenetic tree, which can be specified (TA) or is constructed (GTA) [36, 30, 21, 31]. First the sequences at the leaves are aligned in a pairwise manner. Then in the next step either a representative sequence is calculated or a probabilistic ancestral sequence is determined.

The drawback here is that the correct evolutionary tree is needed, something which is typically not available. We note that the calculation of evolutionary trees is also still an open problem mainly due to the fact that the problem is NP-complete [8].

SP alignment and Maximum Weight Trace: The MSA and maximum weight trace methods [5, 14, 15, 24] differ from the others because they compute an optimal alignment with respect to a well-defined MSA scoring (sum-of-pairs). This is an exact algorithm and the running time is typically $O(k^n)$ which is only practical for a very small number (n) and length (k) of sequences. In addition, the scoring function that is used does not reflect the structure of the underlying evolutionary tree, unless the scoring function is weighted, in which case a tree must be known (or constructed).
A better and more detailed overview of multiple sequence alignment methods is presented in [27, 19, 38].

1.0.6. Error Bounds and Performance Guarantees

There is a bounded-error approximation method by Gusfield [16] for the sum of pairs (SP) measure that calculates an MSA with a score that is never more than twice that of the optimal SP score. Here a scoring method is used where $lower$ scores correspond to better alignments.

Another bound is presented by Bafna, Lawler and Pevzner [2]. Their method aligns n strings and produces an MSA whose SP score can never exceed $2 - \frac{q}{n}$ times the optimal SP score, for any chosen q. For any fixed value of q, the running time of the method is polynomial in k (the assumed string length of each string) and n. As q increases, so does the provable accuracy of the result, but the worst-case running time also increases.

In this extended abstract we present an algorithm that runs in $O(n^2k^2)$ time and produces an MSA whose score is at least $\frac{n-1}{n} \cdot opt$. The score of the alignment represents the probability of the entire evolutionary configuration. In the following sections we briefly explain the scoring function. See [11] for a more detailed exposition. We then describe the algorithm for the calculation of MSAs and present experimental results.

2. Methods

2.0.7. Sum of Pairs versus Circular Sum measure

Definition 2.1 *The tree $T(S) = (V, E, L)$ is a binary, leaf labeled tree with leafset $L = \{s_1, .., s_n\}$.*

In our context a tree $T(S)$ associated with a set of sequences $S = \{s_1, .., s_n\}$ is the tree that corresponds to the evolutionary history of the sequences of S. The internal nodes V represent (usually unknown) ancestor sequences.

To calculate the score with the SP measure [5], all $\binom{n}{2}$ scores of the pairwise alignments are added up. SP methods are obviously deficient from an evolutionary perspective. Consider a tree (Figure 2) constructed for a family containing five proteins. The score of a pairwise alignment $\langle A, B\rangle$ evaluates the probability of evolutionary events on edges (u, A) and (u, B) of the tree; that is, the edges that represent the evolutionary distance between sequence A and sequence B. Likewise, the score of a pairwise alignment $\langle C, D\rangle$ evaluates the likelihood of evolutionary events on edges (C, w), (w, v) and (v, D) of the tree.

By adding "ticks" to the evolutionary tree that are drawn each time an edge is evaluated when calculating the SP score (Figure 2), it is readily seen that with the SP method different edges of the evolutionary tree of the protein family are counted a different numbers of times. In the example tree on the left side that corresponds to the MSA of Figure 1, edges (r, u), (r, w) and (w, v) are each counted six times by the SP method, while edges (u, A), (u, B), (v, D), (v, E), and (w, C) are each counted four times. It gets worse as the tree grows (see tree on the right). There is no theoretical justification to suggest that some evolutionary events are more important than other ones.

Thus, SP methods are intrinsically problematic from an evolutionarily perspective for scoring MSAs. This was the motivation to developed a scoring method that evaluates each edge equally. In addition, we wanted a scoring function that does not depend on the actual tree structure.

2.0.8. Traveling Salesman

Definition 2.2 *A Circular order $C(\mathrm{A})$ of an MSA* A *is any tour through a tree $T(\mathrm{A})$ where each edge is traversed exactly twice, and each leaf is visited once.*

When a tree is traversed in a circular order, that is, from leaf A to B, from B to C, from C to D, from D to E and then back from E to leaf A (Figure 3), all edges are counted

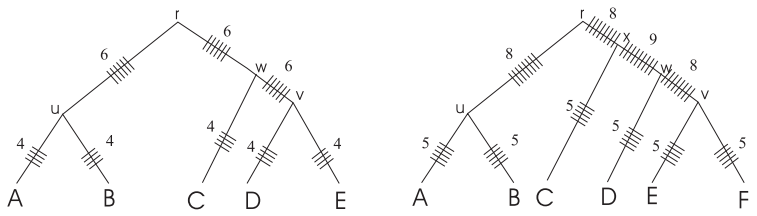


Figure 2. Traversal of trees using the SP measure. Some edges are traversed more often than others.

exactly twice, independent of the tree structure. A circular order is the shortest possible tour through a tree where each leaf is visited once (shortest means smallest sum of edge lengths)(see Figure 3) [11]. Since we do not want to rely on any tree, the problem we consider is to find such an order without having any information about the tree structure.

To solve this problem we reduce it to the symmetric Traveling Salesman Problem (TSP): given is a matrix M that contains the $\binom{n}{2}$ distances of n cities [23, 22]. The problem is to find the shortest tour where each city is visited once. We use a modified version of the problem: in our case, the cities correspond to the sequences and the distances are the scores of the pairwise alignments. In addition, we are interested in the $longest$, not shortest tour, as our scores represent probabilities and we are interested in the $maximum$ probability [1].

Problem 2.1 (TSP problem) *Given is a set of sequences $S = \{s_1, .., s_n\}$ and the corresponding scores of the optimal pairwise alignments. The problem is to find the longest tour where each sequence is visited once.*

Both the construction of optimal evolutionary trees [8, 20] and the Traveling Salesman Problem are NP complete. But the TSP is very well studied and optimal solutions can be calculated within a few hours for up to 1000 cities and in a few seconds for up to 100 cities, whereas the construction of evolutionary trees is still a big problem. There are heuristics for large scale problems that calculate near optimal solutions that are within 1% to 2% of the optimum [29, 13]. For real applications we have seldom more than 100 sequences to compare simultaneously, and the calculation of the optimal TSP solution usually takes only a small fraction of the time it takes to compute all pairwise alignments to derive the scores.

Definition 2.3 *The TSP order $TSP(\mathrm{A})$ of an MSA* A *is the order of the sequences that is derived from the optimal solution of a TSP.*

In summary, we can compute a circular order $C(\mathrm{A})$ without knowing $T(\mathrm{A})$. For a more detailed explanation see [11].

[1] To use any available TSP algorithm, we simply subtract the scores from a large number and then compute the shortest tour.

2.0.9. MSA scoring function

Definition 2.4 *The function $S : \Sigma' \times \Sigma' \rightarrow I\!R$ is defined as follows:*

$$S(x,y) = \begin{cases} DM(x,y), \text{ if } x \neq \text{"_"} \text{ and } y \neq \text{"_"} \\ 0 \text{ if } x = \text{"_"} \text{ and } y = \text{"_"} \\ \text{otherwise an affine gap cost that} \\ \text{depends on the gap length [4]} \end{cases}$$

DM is an entry in the Dayhoff matrix [10].

Our scoring function $F(\mathrm{A})$ of an MSA is similar to the sum-of-pairs measure, except that we do not add all scores of all pairwise alignments, but only the scores of the pairwise alignments in the circular order C divided by two. It is the sum-of-pairs in circular order:

Definition 2.5 *The scoring function $F(\mathrm{A})$ of an MSA $\mathrm{A} = \{a_1, .., a_n\}$ is:*

$$F(\mathrm{A}) = \frac{1}{2} \sum_{m=1}^{k} \sum_{i=1}^{n} S(a_{C_i}[m], a_{C_{i+1}}[m]), \text{ with } C_{n+1} = C_1$$

where k is the length of a sequence in the MSA, n is the number of sequences, C is the TSP ordering, a_{C_i} is a sequence in the MSA, and so the $a_{C_i}[m]$ are amino acids or gap characters within that sequence. $S(x,y)$ is the score of aligning the symbol $x \in \Sigma'$ with the symbol $y \in \Sigma'$. The function $S(x,y)$ is defined above (see Definition 2.4).

2.0.10. Circular Tours and Gaps

Gaps play a special role in MSAs: the process of calculating an MSA can be viewed as inserting gaps at the correct places.

Assume we are given an MSA A and we know the correct tree $T(\mathrm{A})$. Insertions and deletions (indels) are evolutionary events that can be placed in the evolutionary tree (see Figure 5). The result of a deletion event are gaps that appear in all sequences of the subtree where the event took place. In contrast, when the event was an insertion, gaps appear in all sequences that are not in the subtree below the event.

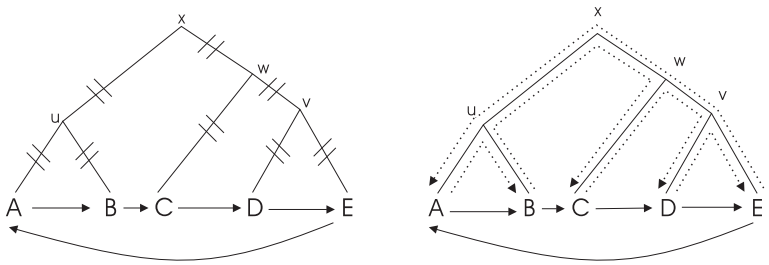


Figure 3. Traversal of a tree in circular order

Definition 2.6 *A gap block B is a set of gaps $\{g_1 \in a_1, .., g_k \in a_k\}$ where the gaps have the following properties: $\forall g_i, g_j \in B : pos(g_i) = pos(g_j)$ and $len(g_i) = len(g_j)$.*

Example: In the alignment of Figure 4, there are 4 indel events and therefore 4 gap blocks. The first gap block consists of two gaps that appear in sequences A and B. We assume we have the correct tree and we also assume that we know the place of the root. In this case the event must be a deletion in the ancestor of sequences A and B. Accordingly, event 2 is an insertion in sequence D, event 3 represents a deletion in the ancestor of D and E, and finally event 4 is an insertion in sequence C.

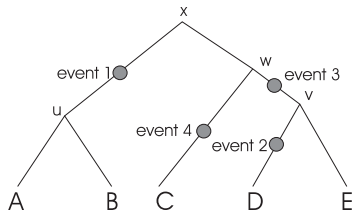


Figure 5. Insertion and deletion events are shown in the evolutionary tree.

The table to the right shows the pairwise scores of the sequences. It is easy to verify that the shortest tour is (A, B, C, D, E, A) and that it is a circular tour in the corresponding tree (see Figure 1). The score of the MSA is the sum of pairwise alignments in circular order divided by two, so $F(\mathrm{A}) = (336 + 79 + 171 + 327 + 110)/2 = 512$. Here the MSA has the optimal score.

Note that the sequences above are ordered in a circular way $C(\mathrm{A})$. Whenever sequences are in a circular order with respect to the evolutionary tree, the gaps that belong to the same evolutionary event appear in a gap block (ignoring the breaking of the circularity).

3. Results

The fact that gaps originating from the same event form gap blocks if a circular tour is known simplifies the task of calculating an MSA.

In Figure 6, a schematic representation of an MSA is shown, where the vertical lines are sequences in the alignment. The rectangles represent gap blocks, and the cylinder itself represents the circular order of the tree.

Once a circular order $C(\mathrm{A})$ is known, we have to select a starting point in the cycle. In principle it does not matter where we start, but as we will see later it is simpler if we cut the cycle in a place where the sequences are farther apart. This is the two sequences in the cycle with the smallest similarity score. We then renumber the sequences with respect to the cycle starting from that point (see Figure 6).

3.0.11. Algorithm

Definition 3.1 *Similar to the definition of the score of an MSA we define the partial score $P(\mathrm{A})$ of an MSA* A*: it is the sum-of-pairs in circular order except for the last term:*

$$P(\mathrm{A}) = \tfrac{1}{2} \sum_{m=1}^{k} \sum_{i=1}^{n-1} S(a_{C_i}[m], a_{C_{i+1}}[m])$$

(see 1.3).

The partial score of the MSA in Figure 4 is $P(\mathrm{A}) = (336 + 79 + 171 + 327)/2 = 456.5$.

Assume we are given n sequences $\{s_1, .., s_n\}$ which are ordered with respect to a circular order C. Assume further that the first k sequences are already aligned and build an MSA $\mathrm{A}' = \langle a_1, .., a_{k-1} \rangle$ with optimal partial score (see Figure 7). We refer to the unaligned input sequences with s_i, to the sequences in the MSA with a_i and to the sequences in an optimal pairwise alignment with t_i.
The algorithm has four steps:

1. take sequence s_k and calculate an optimal pairwise alignment with the next sequence s_{k+1}. The sequences in this alignment with the gaps are called t_k and t_{k+1} (see Figure 8).

```
Event     1          2         3          4
A: RPCVCP___VLRQAAQ__QVLQRQIIQGPQQLRRLF_AA
B: RPCACP___VLRQVVQ__QALQRQIIQGPQQLRRLF_AA
C: KPCLCPKQAAVKQAAH__QQLYQGQLQGPKQVRRAFRLL
D: KPCVCPRQLVLRQAAHLAQQLYQGQ____RQVRRAF_VA
E: KPCVCPRQLVLRQAAH__QQLYQGQ____RQVRRLF_AA
```

	A	B	C	D	E
A	0	336	27	44	110
B	336	0	79	56	99
C	27	79	0	171	176
D	44	56	171	0	327
E	110	99	176	327	0

Figure 4. An MSA and a table containing the pairwise scores

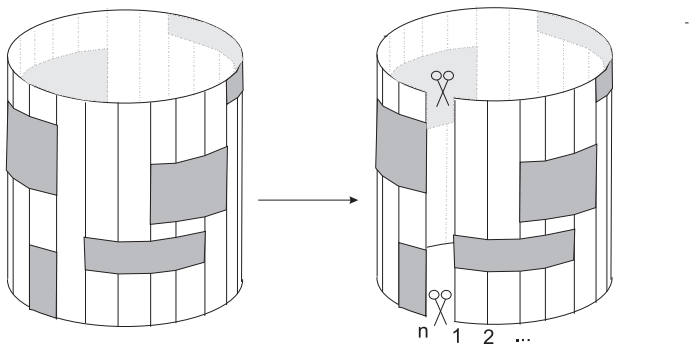


Figure 6. Schematic representation of an MSA in a circular order

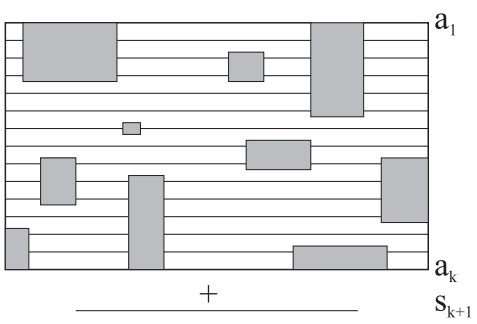


Figure 7. Sequences $\langle a_1, .., a_k \rangle$ are aligned and build an MSA A' with optimal partial score.

2. insert all gaps from t_k that are not already present in a_k into all previously aligned sequences $\langle a_1, .., a_k \rangle$ (see Figure 9).

3. take all the gaps that were present in a_k before step 2), and insert them in both, t_k and t_{k+1}, except for the gaps that are already present in t_k (see Figure 9).

4. at this point $a_k = t_k$ and $a_{k+1} = t_{k+1}$. We add a_{k+1} to the alignment and increment k by one (see Figure 9).

We now discuss steps 2 to 4 and show that in each step the MSA has an optimal partial score:

Step 2 We only insert those gaps of t_k into the MSA that are not $already$ present in a_k. Also, when a gap in t_k overlaps with a gap in a_k, we only insert the the part of the gap that is not already present in a_k (see Figure 9, large X).

Since each gap of t_k that is considered is inserted into all sequences $\langle a_1, .., a_k \rangle$, the partial score $P(\mathrm{A})$ is still optimal: gap against gap scores zero. In Figure 9, the up arrow correspond to this step. In the example one gap is inserted into the MSA (tall dark rectangle). The large "X" marks the gap that is already present and is therefore not inserted.

Step 3 Again, only those gaps of a_k are inserted into t_k and t_{k+1} that are not already present in t_k.

Since each gap of sequence a_k that is considered is inserted into $both$, t_k and t_{k+1}, the score of this alignment is still optimal, as gap against gap scores zero. In Figure 9 the down arrows correspond to this step. The dark rectangles represent the inserted gaps.

After those two steps the partial score of the MSA and the score between sequences a_k and a_{k+1} are both still optimal.

Step 4 Since now each gap of t_k is also in a_k at the same position, and each gap of a_k is in t_k, a_k is equal to t_k. This means that we can simply add t_{k+1} to the MSA and rename t_{k+1} to a_{k+1}.

The last step We continue this process until all n sequences are in the MSA. We have shown that the partial score of the MSA is optimal. The score $F(\mathrm{A})$ differs from the partial score only in the last term in the sum. The last term comes from scoring the last sequence a_n with the first

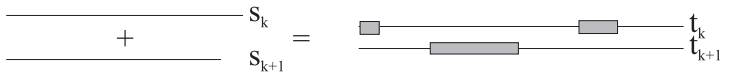


Figure 8. Step 1:

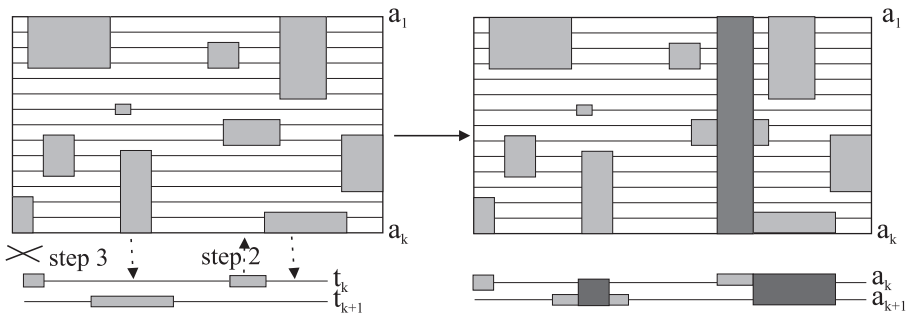


Figure 9. Steps 2 and 3: all gaps from t_k are inserted into the MSA, and all gaps from a_k are inserted into t_k and t_{k+1}

sequence a_1 in the MSA. The problem is that in the algorithm we never aligned s_n with s_1 to obtain an optimal pairwise alignment, hence this score may not be optimal.

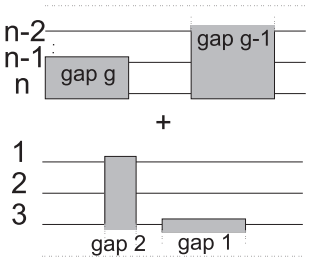


Figure 10. The alignment between sequences a_1 and a_n might not be optimal

3.0.12. Lower bound

Our algorithm guarantees a lower bound $L(\mathrm{A})$ of least $\frac{n-1}{n} \cdot opt$ (where opt is the maximum possible score): $L(\mathrm{A}) \geq \frac{n-1}{n} \cdot opt$

The reason why we choose a place to cut the circle where the sequences are distant is the following: the score of the pairwise alignment that has the smallest score (in $C(\mathrm{A})$) adds the smallest amount to the overall sum. Since this is the only pairwise alignment that is not optimal within the MSA, we get the greatest possible lower bound $L(\mathrm{A})$. Hence the lower bound is $L(\mathrm{A}) = \frac{n-1}{n} \cdot opt$ when all the pairwise scores are the same, and it is greater otherwise.

Our algorithm so far had to invest $O(nk^2)$ time for the circular alignment (where k is the length of the longest sequence) and $O(n^2k^2)$ time to compute all pairwise alignments at the beginning, which is the most expensive part. Of course the TSP is exponential in n, but in real cases the calculation of the TSP order takes only a fraction of the computation time compared to the time it takes to construct the score matrix of the $\binom{n}{2}$ pairwise alignments.

3.0.13. Further improvements

At this point we could consider the following possibilities:

- Do nothing: in most cases the score is close enough to the optimal that we do not need to make any adjustments.
- Use a heuristic that shifts the gap blocks around to come closer to the optimum.
- Use an exhaustive algorithm to shift gap blocks around.

If we want to optimize this alignment the first thing we consider is: whatever we change, the score between the first $n-1$ alignments within the MSA can only get $worse$, since they are optimal. The only possible improvement would be to increase the score of the $last$ alignment more than to decrease the scores of all the other alignments. This makes the search for an optimal alignment simple: we only have to consider alignments in the search where the last pairwise alignment n increases, and discard all other cases.

We show that it is very unlikely that we have to insert a gap anywhere in order to improve the alignment. Since all the pairwise alignments within the MSA are optimal except

for the alignment of the last with the first sequence, any gaps we insert will decrease the score. So the only possible place where we could insert a gap in order to increase the score is either in sequence a_1 or a_n (and not both, as this would not affect the alignment $\langle a_1, a_n \rangle$):

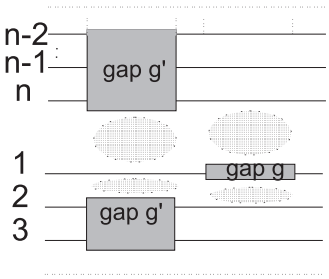


Figure 11. Insertion of gap g in sequence a_1 or a_n results in an insertion of gaps in all other sequences. The score of the MSA is affected at four places (clouds)

But if we insert a gap g in either sequence a_n or a_1, we have to insert another gap g' in all other sequences, otherwise some sequences in the MSA would be longer than others (see Figure 11). The score of the previously optimal alignment $\langle a_1, a_2 \rangle$ is now lower by the cost of two gaps (see Figure 11). In addition, the alignment $\langle a_n, a_1 \rangle$ has two additional gaps. Hence the score of the MSA could only improve, if the alignment $\langle a_1, a_n \rangle$ has now a score that is at least four times a gap cost greater than it was before.

Note that the gap cost is usually very low compared to the score of aligning two amino acids. In our case the fixed gaps cost is around -14, and the score of aligning two Ala is 2.4 (depending on the PAM distance) [6, 10]. So it is very unlikely that the alignment $\langle a_1, a_n \rangle$ gets better after inserting two gaps. In addition, this situation is very easy to check: we simply subtract the score of the alignment $\langle a_1, a_n \rangle$ from the *optimal* score $OPA(s_1, s_n)$. If the difference is smaller than the cost of four gaps, we know for sure that we do not have to insert a gap - otherwise the alignment $\langle a_1, a_n \rangle$ could not improve by four gaps.

3.0.14. Optimizing the border blocks

First we need to consider where the score changes when gap blocks are moved. Let us look at the gap block with the label $gap\ g$ as an example in Figure 12. If we move this gap block to the right, the score of the overall alignment is affected at exactly two places: the score between the pairwise alignment $\langle a_{n-2}, a_{n-1} \rangle$ changes, and the score on the other side of the gap block of the alignment $\langle a_n, a_1 \rangle$ changes. The rest of the alignment is not disturbed. This also means that the gap blocks do not influence each other, unless they end at the same place.

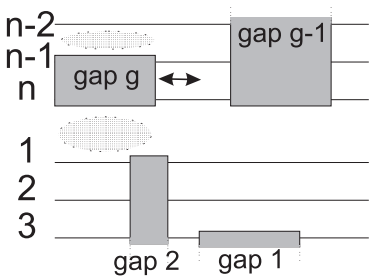


Figure 12. The movement of a gap block only affects two places in the alignment.

We call gap blocks that end in either sequence a_1 or a_n border blocks.

The only way to improve the $last$ pairwise alignment is to move border blocks around. But whenever we move a gap block, we also disturb one optimal alignment somewhere between sequences a_1 and a_n where the border block ends. But it is possible that moving gap border blocks around will increase the score of the last alignment more than it will decrease the overall score of the alignment. The most promising optimization to do would be to remove a border block by joining border blocks that are cut apart like the ones in Figure 13. This way we remove two times the cost of a gap for each border block, and at the same time the optimal alignments of the neighboring sequences are only disturbed at one place per block (see Figure 13). To do this we only consider border blocks of same size, because if we take border blocks of different size, there would still be a gap left.

If there are several bordering gap blocks of same size we need to invest some brute force searching, which is exponential in the number of bordering gap blocks: if there are b bordering gap blocks it would take in the order of $O(k^b)$ time to compute all possibilities.

However, in real cases, it does not make sense to move one gap block from one end to the complete other end of the alignment, if we assume that most gap blocks are more or less placed correctly within a certain distance. So using heuristics this complexity can further be reduced. But even without the above optimization we already have an MSA close to the optimum.

4. Example

To show a real example we took the tat protein from several strains of the Human Immunodeficiency Virus type 1 (HIV-1). The first alignment was calculated using Clustal W [35], the second one using our new algorithm, including shifting of the border blocks. Due to the size of the MSAs,

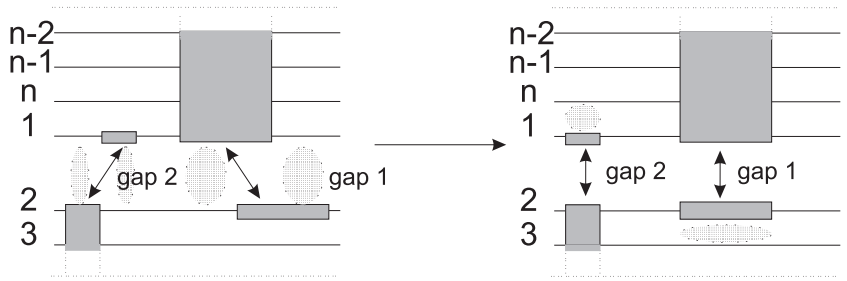


Figure 13. Before the optimization there are four border blocks. After shifting, two optimal alignments are disturbed a bit, but in exchange two gap blocks are gone.

```
Clustal W:
---------
Score of the alignment: 10968.663
Maximum possible score: 11070.457

1    ____MDPVDPNLESWNHP__________________GS_______QPRTACNK_CHCKKCCYHC
2    ____MDPVDPNLEPWNHP__________________GS_______QPRTPCNK_CHCKKCCYHC
9    ____MEPVDPRLEPWKHP__________________GS_______QPKTACNN_CYCKKCCYHC
16   ____MEPVDPRLEPWKHP__________________GS_______QPKTASNN_CYCKRCCLHC
20   ____MEPVDPRLEPWKHP__________________GS_______QPKTACTN_CYCKKCCFHC
22   ____MEPVDPRLEPWKHP__________________GS_______QPKTACTT_CYCKKCCFHC
21   ____MDPVDPRLEPWKHP__________________GS_______QPKAACTS_CYCKKCCFHC
23   ____MEPVDPSLEPWKHP__________________GS_______QPKTACTN_CYCKKCCLHC
18   ____MEPVDPNLEPWKHP__________________GS_______QPRTACNN_CYCKKCCFHC
19   ____MEPVDPNLEPWKHP__________________GS_______QPRTACNN_CYCKKCCFHC
8    ____MEPVDPNLEPWKHP__________________GS_______QPRTACTN_CYCKKCCFHC
15   ____MEPVDPRLEPWKHP__________________GS_______QPKTACTN_CYCKKCCFHC
14   ____MEPVDPRLEPWKHP__________________GS_______QPKTACTN_CYCKKCCFHC
13   ____MEPVDPRLEPWKHP__________________GS_______QPKTACTN_CYCKKCCFHC
12   ____MEPVDPRLEPWKHP__________________GS_______QPKTACTN_CYCKKCCFHC
11   ____MEPVDPRLEPWKHP__________________GS_______QPKTACTN_CYCKKCCFHC
10   ____MEPVDPRLEPWKHP__________________GS_______QPKTACTT_CYCKKCCFHC
17   METHLKAPESSLESYNEPSSCTSEQGVTAQELAKQGEELLSQLHRPLEACTNSCYCKQCSFHC
7    ____MEPVDPNLEPWKHP__________________GS_______QPTTACSN_CYCKVCCWHC
6    ____MDPIDPDLEPWKHP__________________GS_______QPRTVCNN_CYCKACCYHC
24   ____MDPVDPNIEPWNHP__________________GS_______QPKTACNR_CHCKKCCYHC
5    ____MDPVDPNIEPWNHP__________________GS_______QPKTACNR_CHCKKCCYHC
4    ____MDPVDPNLEPWNHP__________________GS_______QPKTACNR_CHCKKCCYHC
3    ____MDPVDPNLEPWNHP__________________GS_______QPRTPCNK_CYCKKCCYHC

 1 ->  P18804      2 ->  P04611      3 ->  P04613      4 ->  P04609
 5 ->  P12506      6 ->  P17285      7 ->  P24738      8 ->  P19552
 9 ->  P05908     10 ->  P04610     11 ->  P04607     12 ->  P04606
13 ->  P04326     14 ->  P04608     15 ->  P05907     16 ->  P20893
17 ->  P18044     18 ->  P35965     19 ->  P04614     20 ->  P19553
21 ->  P05906     22 ->  P05905     23 ->  P20879     24 -> PDB1tiv
```

```
Circular pairwise alignment:
----------------------------
Score of the alignment: 10995.831
Maximum possible score: 11070.457

1    ____MDPVDPNLESWNHPGS_________________________QPRTACNK_CHCKKCCYHC
2    ____MDPVDPNLEPWNHPGS_________________________QPRTPCNK_CHCKKCCYHC
9    ____MEPVDPRLEPWKHPGS_________________________QPKTACNN_CYCKKCCYHC
16   ____MEPVDPRLEPWKHPGS_________________________QPKTASNN_CYCKRCCLHC
20   ____MEPVDPRLEPWKHPGS_________________________QPKTACTN_CYCKKCCFHC
22   ____MEPVDPRLEPWKHPGS_________________________QPKTACTT_CYCKKCCFHC
21   ____MDPVDPRLEPWKHPGS_________________________QPKAACTS_CYCKKCCFHC
23   ____MEPVDPSLEPWKHPGS_________________________QPKTACTN_CYCKKCCLHC
18   ____MEPVDPNLEPWKHPGS_________________________QPRTACNN_CYCKKCCFHC
19   ____MEPVDPNLEPWKHPGS_________________________QPRTACNN_CYCKKCCFHC
8    ____MEPVDPNLEPWKHPGS_________________________QPRTACTN_CYCKKCCFHC
15   ____MEPVDPRLEPWKHPGS_________________________QPKTACTN_CYCKKCCFHC
14   ____MEPVDPRLEPWKHPGS_________________________QPKTACTN_CYCKKCCFHC
13   ____MEPVDPRLEPWKHPGS_________________________QPKTACTN_CYCKKCCFHC
12   ____MEPVDPRLEPWKHPGS_________________________QPKTACTN_CYCKKCCFHC
11   ____MEPVDPRLEPWKHPGS_________________________QPKTACTN_CYCKKCCFHC
10   ____MEPVDPRLEPWKHPGS_________________________QPKTACTT_CYCKKCCFHC
17   METHLKAPESSLESYNEPSSCTSEQGVTAQELAKQGEELLSQLHRPLEACTNSCYCKQCSFHC
7    ____MEPVDPNLEPWKHPGS_________________________QPTTACSN_CYCKVCCWHC
6    ____MDPIDPDLEPWKHPGS_________________________QPRTVCNN_CYCKACCYHC
24   ____MDPVDPNIEPWNHPGS_________________________QPKTACNR_CHCKKCCYHC
5    ____MDPVDPNIEPWNHPGS_________________________QPKTACNR_CHCKKCCYHC
4    ____MDPVDPNLEPWNHPGS_________________________QPKTACNR_CHCKKCCYHC
3    ____MDPVDPNLEPWNHPGS_________________________QPRTPCNK_CYCKKCCYHC

 1 ->  P18804      2 ->  P04611      3 ->  P04613      4 ->  P04609
 5 ->  P12506      6 ->  P17285      7 ->  P24738      8 ->  P19552
 9 ->  P05908     10 ->  P04610     11 ->  P04607     12 ->  P04606
13 ->  P04326     14 ->  P04608     15 ->  P05907     16 ->  P20893
17 ->  P18044     18 ->  P35965     19 ->  P04614     20 ->  P19553
21 ->  P05906     22 ->  P05905     23 ->  P20879     24 -> PDB1tiv
```

only a part is shown where the results obtained by the two methods differ.

The scores are determined using the CS measure. The maximum possible score in the example is the score determined using the CS measure but using the OPAs: it is the sum of the OPAs in circular order (the upper bound for the MSA A, see [11]).

The score obtained with the new algorithm is slightly better than the score obtained from Clustal W. It is clearly better at the beginning (if you ask an expert biologist), because the alignment has only one big gap whereas the alignment obtained from Clustal W has two gaps.

With the above example we do not want to prove that our method performs generally better than Clustal W, but we want to show that our method is usable and produces good results.

5. Discussion

Our algorithm produces an MSA that has a score of at least $\frac{n-1}{n} \cdot opt$, where opt is the maximum possible score. It takes in the order $O(n^2k^2)$ time given a circular tour. The calculation of such a tour using a TSP algorithm can be done very efficiently, and in realistic cases the calculation time of a TSP cycle can be ignored. The alignment can then further be improved using an exact or heuristic algorithm. A further advantage of this algorithm is that it does not need to calculate an evolutionary tree but uses a scoring function that takes such a tree into account.

6. Acknowledgements

We would like to thank Todd Wareham for various references and Ulrike Stege for reading the paper.

References

[1] S. Altschul and B. W. Erickson. Optimal sequence alignment using affine gap costs. *J. Mol. Biol.*, 48:603 –1 6, 1986.

[2] V. Bafna, E. Lawler, and P. Pevzner. Approximation algorithms for multiple sequence alignment. *Proc. 5th Symp. on Combinatorial Pattern Matching*, pages 43 – 53, 1994.

[3] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. A. McClure. Hidden markov models of biological primary sequence information. *Proc. Natl. Acad. Sci. USA*, 91:1059–1063, 1994.

[4] S. A. Benner, M. A. Cohen, and G. H. Gonnet. Empirical and structural models for insertions and deletions in the divergent evolution of proteins. *J. Molecular Biology*, 229:1065–1082, 1993.

[5] H. Carillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.*, 48(5):1073–1082, 1988.

[6] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model for evolutionary change in proteins. In M. O. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, volume 5, pages 345–352. 1978.

[7] D. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, 25:351 – 60, 1987.

[8] L. R. Foulds and R. L. Graham. The steiner problem in phylogeny is np-complete. *Proc. Natl. Academy Science*, 3:43 – 49, 1982.

[9] G. H. Gonnet. A tutorial introduction to computational biochemistry using Darwin. 1994.

[10] G. H. Gonnet, M. A. Cohen, and S. A. Benner. Exhaustive matching of the entire protein sequence database. *Science*, 256:1443–1445, 1992.

[11] G. H. Gonnet and C. Korostensky. Evaluation measures of multiple sequence alignments. *J. Comp. Biol.*, 1999. submitted.

[12] O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162:705–708, 1982.

[13] M. Groetschel and O. Holland. Solution of large-scale symmetric traveling salesmand problems. *Math. Programming*, pages 141 – 202, 1991.

[14] S. Gupta, J. Kececioglu, and A. Schaffer. Making the shortest-paths approach to sum-of-pairs multiple sequence alignment more space efficient in practice. *Proc. 6th Symp. on Combinatorial Pattern Matching*, pages 128 – 43, 1995.

[15] S. K. Gupta, J. Kececioglu, and A. A. Schaffer. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. In *J. Computational Biology*, 1996.

[16] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bull. Math. Biol.*, 55:141 – 54, 1993.

[17] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Academy Science*, 89:10915 – 19, 1992.

[18] S. Henikoff and J. G. Henikoff. Blocks database and its applicatioins. In R. F. Doolittle, editor, *Methods in Enzymology*, volume 266 of *Computer methods for macromolecular sequence analysis*, pages 88 – 105. Academic Press, New York, 1996.

[19] M. Hirosawa, Y. Totoki, M. Hoshida, and M. Ishikawa. Comprehensive study on iterative algorithms of multiple sequence alignment. *CABIOS*, 11(1):13 – 18, 1995.

[20] T. Jiang and L. Wang. On the complexity of multiple sequence alignment. *J. Comp. Biol.*, 1:337 – 48, 1994.

[21] T. Jiang, L. Wang, and E. L. Lawler. Approximation algorithms for tree alignment with a given phylogeny. *Algorithmica*, 16:302 – 15, 1996.

[22] D. Johnson. More approaches to the travelling salesman guide. *Nature*, 330:525, December 1987.

[23] D. Johnson. Local optimization and the traveling salesman problem. In *Proc. 17th Colloq. on Automata, Languages and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 446 – 461, Berlin, 1990. Springer Verlag.

[24] J. Kececioglu. The maximum weight trace problem in multiple sequence alignment. *Proc. 4th Symp. on Combinatorial Pattern Matching*, pages 106 – 19, 1993.

[25] A. Krogh, M. Brown, I. S. Mian, K. Sjolander, and D. Haussler. Hidden markov models in computational biology: Applications to protein modeling. *J. Molecular Biology*, 235:1501–1531, 1994.

[26] H. Martinez. A flexible multiple sequence alignment program. *Nucleic Acids Res.*, 16:1683 – 1691, 1988.

[27] M. McClure, T. Vasi, and W. Fitch. Comparative analysis of mutliple protein sequence alignment methods. *J. Mol. Biol. Evol*, 11(4):571 – 592, 1994.

[28] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[29] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33:60 – 100, 1991.

[30] R. Roui and J. Kececioglu. Approximation algorithms for multiple sequence alignments under a fixed evolutionary tree. *Discrete Applied Mathematics*, pages 355 – 366, 1998.

[31] D. Sankoff and R. Cedergren. Simultaneous comparison of tree or more sequences related by a tree. In D. Sankoff and G. Kruskal, editors, *Time Warps, String Edits, and Marcomolecules: the Theory and Practice of Seqeunce Comparison*, volume 28, pages 253 – 263. Addison Wesley, Reading MA, 1983.

[32] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

[33] W. Taylor. Multiple sequence alignment by a pairwise algorithm. *Comput. Appl. Biosci.*, 3:81 –87, 1987.

[34] W. Taylor. A flexible method to align a large number of sequences. *J. Mol. Evol.*, 28:161 – 169, 1988.

[35] J. Thompson, D. Higgins, and T. Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.

[36] L. Wang and D. Gusfield. Improved approximation algorithms for tree alignment. *Proc. 7th Symp. on Combinatorial Pattern Matching*, pages 220 – 33, 1996.

[37] M. Waterman and M. Perlwitz. Line geometries for sequence comparison. *Bull. Math. Biol.*, 46:567 – 577, 1984.

[38] A. Wong, S. Chan, and D. Chiu. A multiple sequence comparison method. *Society for Mathematical Biology*, 55(2):465–486, 1993.